Journal of Algorithms and Computation

journal homepage: http://jac.ut.ac.ir

Online Scheduling of Jobs for D-benevolent instances On Identical Machines

I. Mohammadi [*1] and D. Moazzami [†2]

[1]University of Tehran, Department of Algorithms and Computation.
[2]University of Tehran, College of Engineering, Faculty of Engineering Science

ABSTRACT

We consider online scheduling of jobs with specific release time on m identical machines. Each job has a weight and a size; the goal is maximizing total weight of completed jobs. At release time of a job it must immediately be scheduled on a machine or it will be rejected. It is also allowed during execution of a job to preempt it; however, it will be lost and only weight of completed jobs contribute on profit of the algorithm. In this paper we study D-benevolent instances which is a wide and standard class and we give a new algorithm, that admits $(2m + 4)$-competitive ratio. It is almost half of the previous known upper bound for this problem.

ARTICLE INFO

Article history:
Received 20, March 2015
Received in revised form 12, January 2016
Accepted 3, March 2016
Available online 24, March 2016

Keyword: Online Algorithms Scheduling Identical Machine Upper bound

AMS subject Classification: Primary 05C80

1 Introduction

*Email:*mohammadi.iman@alumni.ut.ac.ir*
†E-mail: *dmoazzami@ut.ac.ir*

We deal with online scheduling of problems with fixed start time to maximize total weight of jobs. It is a well-studied problem with many applications. For instance call control and bandwidth allocation in communication channels [1, 2]. Each job has a size and a weight associated with. In this paper we consider it for identical machines. Jobs are released online with fixed start time. On arrival of jobs, they are scheduled on m machines, unlesss they will be rejected. The number of jobs and future release times are unknown. Preemption is allowed, which means every job can be preempted at any time; however, preempted jobs will be lost. Preemption can be useful, when more valuable jobs are released and there are no available machines to schedule them (it is said that a machine is available, if it is idle or just finished execution of a job). Without preemption, it is easy to see that no online algorithm can be competitive for most models. However, power of preemption is bounded and in some cases preemption does not improve algorithms [2].

An online algorithm is called R-competitive if the total weight of completed jobs is at least $1/R$ as the optimum schedule for any instance. In the most general setting, no algorithm has bounded competitive ratio (Section 1.1) and we consider a standard case, D-benevolent instances which is a wide class of instances. In D-benevolent instances weight of job is determined by a fixed function f of its size. A function f is D-benevolent if it is decreasing on $(0, \infty)$, $f(0) = 0$, and $f(p) > 0$ for all $p > 0$. (Hence such functions have a discontinuity at 0.)

1.1 Previous Result

As mentioned, in the general case, where jobs can have arbitrary weights and size, there is no any online (randomized) algorithm with competitive ratio even on single machine [7, 2]. [7] gave optimal 4-competitive algorithms for unit sized jobs with arbitrary weights, D-benevolent jobs, and C-benevolent jobs, a single machine. The lower bound of 3 for all surjective functions (includes D-benevolent instances) and single machine was shown by the same author [7]. [5] and [3] considered the version of jobs with unit weights (a special case of D-benevolent) on m identical machines. They gave a 1-competitive algorithm for this problem. The lower bound of 2 in randomized algorithms was improved for unit size, C-benevolent and D-benevolent instances by [6].

[4] gave a fact that "*If algorithm Alg is R-competitive on a single machine, then an algorithm that uses only the fastest machine by simulating Alg on it is $(R.m)$-competitive on m related machines.*". Therefore, by the fact, they extended [7] results (4-competitive algorithm on single machine for D-benevolent instances) to $4m$ upper bound on m related machines for D-benevolent instances which implies on identical machines. They also gave

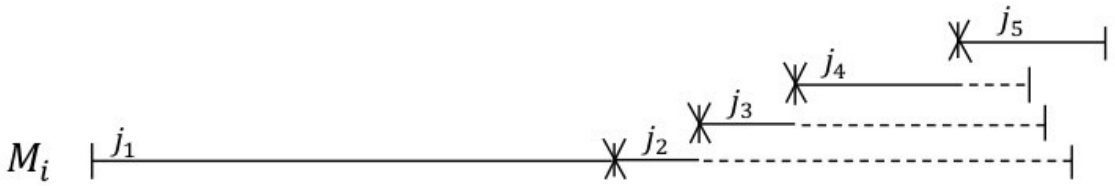

Figure 1: Execution of chain $\{j_1, j_2, j_3, j_4, j_5\}$ by M_i, sequence of jobs $\{j_2, j_3, j_4\}$ is an only heap in this chain and set H_4 contains jobs $\{j_2, j_3\}$ and j_5 is only job in the chain completed by M_i

lower bound m for instances with unit-weight variable-sized jobs that are a special case of D-benevolent instances, thus it holds the lower bound for D-benevolent instances on related machines.

1.2 Our Result

In this paper we consider D-benevolent instances on m identical machines and give a new algorithm. As mentioned, the previous upper bound for this problem is $4m$ and lower bound is unknown which is a big gap [4]. Our algorithm improves the upper bound and admit $(2m + 4)$-competitive ratio.

We also prove that our algorithm holds [7] upper bound in case of single machine with 4-competitive ratio. Therefore our algorithm admits best known upper bound on m identical machines.

2 Notations and Definitions

We consider online scheduling of jobs $\{j_1, \dots\}$ on $m \geq 1$ identical machines. All Machines have unit speed. $r(j)$, $p(j)$ and $w(j) > 0$ denotes release time, size and weight of job j, respectively. $[r(j), d(j))$ is a time interval in which an algorithm runs job j and $d(j)$ is completed or preempted time of job j. Consider job j executing on machine M_i, if j is completed, $d(j) = r(j) + p(j)$ and if j is preempted, $d(j) < r(j) + p(j)$. We call a machine is idle if it is not running any job, otherwise it is busy.

For a given algorithm ALG and instance I, $ALG(I)$ and $OPT(I)$ denotes total weight of completed jobs scheduled by ALG and optimal solution. ALG is called R-competitive if $OPT(I) \leq R.ALG(I)$ for any instance I.

Definition 1. *A chain is a maximal sequence of jobs $j_1, \ldots, j_n$ running on one machine, in which j_k is preempted when j_{k+1} arrives $(k = 1 \ldots n-1)$.*

Definition 2. *A heap $j_l, \ldots, j_h$ is a maximal sub sequence of a chain, in which $r(j_k) + p(j_k) \geq r(j_{k+1}) + p(j_{k+1}) \quad (k = l \ldots h-1)$ (it is called heap since figurative shape of the sequence looks like a heap of jobs, e.g. $\{j_2, j_3, j_4\}$ in Fig. 1).*

Definition 3. *Consider chain $j_1, \ldots, j_n$, H_k for $1 \leq k \leq n$ is a set of jobs j_d in which $d \leq k$ and j_d is within a heap but not the last job of a heap.In the other words, H_k is a collection of heaps which are appeared before k^{th} job of the chain with the exception of the last job of the heaps. Moreover, $w(H_k)$ denotes total weight of jobs in H_k.*

Due to the fact that every job is within a chain, there are two states for a machine. Either a machine is idle or is busy and running a chain. At most m chain is running at same time and a chain contains at least one job and at most all of the jobs. Also chain may contain zero or more heaps.

Set H_k contains all the jobs in H_{k-1} and probably j_k, refer to the definition 3 (**Note:** during execution of j_k, H_k could not contain j_k; however, it may be contained later, by releasing future jobs and extending the heap in which j_k is contained).

Fig. 1 illustrates execution of chain $\{j_1, j_2, j_3, j_4, j_5\}$ by machine M_i. Last job of the chain is only job in the chain that is completed and other jobs are preempted by the next arrived job. Sequence $\{j_2, j_3, j_4\}$ is only heap within the chain, thus we have $H_5 = \{j_2, j_3\}, H_4 = \{j_2, j_3\}, H_3 = \{j_2, j_3\}, H_2 = \{j_2\}, H_1 = \{\}$.

3 Linear Competitive Algorithm for D-benevolent Instances

In this section we introduce an algorithm ALG that schedules D-benevolent instances with 4-competitive ratio on single machine and $(2m + 4)$-competitive ratio on $m > 1$ machines. The weight of jobs in this class of input are given by a D-benevolent function f of their sizes, that is, $w(j) = f(p(j))$. On arrival of a new job j, algorithm ALG decides to schedule j or reject it.

***Algorithm** ALG:*

1. If there are any idle machines, run j on one of them.

2. Otherwise, all machines are busy, between m chains running on machines find chain $\{j_1, \ldots, j_k\}$ in which $[(\sum_{d=1}^{k} w(j_d)) - w(H_k)] < w(j)$, then preempt j_k and run j on the same machine.

3. Otherwise, for a job j' running on machine M_i that $1 \leq i \leq m$ and $r(j') + p(j') \geq r(j) + p(j)$, preempt j' and run j on the same machine, if exists.

4. Otherwise, reject j.

Observation 1. *For a chain $\{j_1, \ldots, j_n\}$ executed on machine i by ALG, M_i is idle when j_1 arrives and j_n is only job in the chain that is completed. During $[r(j_k), d(j_k))$ ALG runs j_k , in which $d(j_n) = r(j_n) + p(j_n)$ and $d(j_k) = r(j_{k+1}) < r(j_k) + p(j_k)$ for $k = 1 \ldots n-1$.*

Observation 2. *Consider a chain $\{j_1, \ldots, j_n\}$ executed by ALG, we have $w(j_k) > [(\sum_{d=1}^{k-1} w(j_d)) - w(H_{k-1})]$ and $[(\sum_{d=1}^{k} w(j_d)) - w(H_k)] \leq 2w(j_k)$ for $k = 1 \ldots n$.*

Observation 3. *OPT completes all of it's selected jobs. Consider job j' that is run by OPT on M_i. It would be said that j' is associated with a specific chain run by ALG if the chain was running on the same machine at the same time. Likewise, It would be said that j' is associated with job j run by ALG if j is running on M_i and $r(j') \in [r(j), d(j))$.*

3.1 Upper bound of ALG for single machine

Consider the problem for single machine; in this case we give upper bound 4 for *ALG*.

Claim 1. *For $m = 1$, every job in OPT is associated with a chain in ALG.*

Proof. Assume that j' run by OPT, is not associated with any chain in ALG, thus during release time of j', the single machine is idle in ALG. Therefore, according to step 1, j' would be scheduled in ALG and it becomes first job of a chain with which j' is also associated. A contradiction. □

Lemma 1. *For $m = 1$, consider a chain $\{j_1, \ldots, j_n\}$ containing a heap $\{j_l, \ldots, j_h\}$, OPT starts to run at most one job j' in time interval $[r(j_l), d(j_h))$ and also $r(j') \geq r(j_h)$.*

Proof. There is no job that is released after $r(j_l)$ and completed before $r(j_h) + p(j_h) \geq d(j_h)$. Otherwise, $\{j_l, \ldots, j_h\}$ could not be a heap according to definition 2 and step 3 of ALG, which means the existed job would be contained into the heap as a last job of the heap. Therefore, every job which OPT starts to run during $[r(j_l), d(j_h))$ would be completed in time $t \geq d(j_h)$ thus only one job j' can be executed by OPT (because of overlapping at $d(j_h)$, it is impossible to run more than one job). In addition, we have $r(j') \geq r(j_h)$; otherwise we would have $p(j') > p(j_h) \Rightarrow w(j') < w(j_h)$ which is a contradiction with optimal solution. □

Theorem 1. *For $m = 1$, the competitive ratio of ALG is at most 4 for D-benevolent instances.*

Proof. Consider OPT runs maximal set of jobs $\{j'_1, \ldots, j'_{n'}\}$ which are associated with a specific chain $\{j_1, \ldots, j_n\}$ in ALG, it is sufficient to compare total weight of jobs $\{j'_1, \ldots, j'_{n'}\}$ with $w(j_n)$ (The only job in the chain that ALG completes). Now imagine $j_k (k = 1 \ldots n)$ is in the chain run by ALG and we have $r(j'_f) \in [r(j_k), d(j_k))$ in which j'_f is run by OPT $(f = 1 \ldots n')$. It is clear that $w(j'_f) \leq [(\sum_{d=1}^{k} w(j_d)) - w(H_k)] \leq 2w(j_k)$; otherwise, by step 2 of ALG, j'_f would preempt j_k. In that case we would have $j'_f = j_{k+1}$, which contradicts with assumption of release time of j'_f. Further more, j'_f is only job started to run by OPT in $[r(j_k), d(j_k))$. If it was not, there would be some jobs started and completed within this time interval, thus by step 3 of ALG, first of them would be scheduled on the machine and it would become j_{k+1}, that is a contradiction with assumption of $r(j_{k+1}) = d(j_k)$. In addition, by lemma 1, no jobs are started to run by OPT while a job in H_n is running by ALG.
Hence:

$$\sum_{d=1}^{n'} w(j'_d) \leq 2 \left[\left(\sum_{k=1}^{n} w(j_k) \right) - w\left(H_n\right) \right] \leq 4w(j_n)$$

Note: the second inequality directly comes from observation 2 □

3.2 Upper bound of ALG for parallel identical machines

Consider the problem for m parallel identical machines; in this case we give upper bound $(2m + 4)$ for ALG.

Claim 2. *For $m > 1$, consider job j' only executed by OPT and not by ALG. In this case, j' would certainly be associated with a chain in ALG.*

Proof. Assume to the contrary that j' is not associated with any chain in ALG, due to step 1, ALG would run j' on an idle machine, a contradiction. □

We assign weight of every job executed by OPT to either a job or a chain executed by ALG. Consider job j' that OPT runs on machine i, the assignment rules is defined as follows.

1. If ALG does not run j', by claim 2 there is a chain with which j' is associated, assign $w(j')$ to the chain.

2. Otherwise, assign $w(j')$ to itself where ALG runs it.

Now we can compute the total weight assigned to a specific chain and jobs of it.

Lemma 2. *Assume ALG runs a heap $\{j_l, \ldots, j_h\}$ on M_i , OPT starts to run at most one job j' in time interval $[r(j_l), d(j_h))$ on M_i , such that j' is not run by ALG.*

Proof. Consider OPT runs a sequence of jobs $j'_1, \ldots j'_f$ within $[r(j_l), d(j_h))$, we show j'_f is only job that may not be run by ALG. We know in the heap $r(j_d) + p(j_d) \geq d(j_h)$ for $l \leq d \leq h$ and OPT completes all of its' selected jobs, so $r(j'_z) + p(j'_z) < r(j'_f) < d(j_h)$ for $z = 1 \ldots f-1$. Hence, according to step 3 of ALG, there is at least one step in ALG to run j'_z. □

Lemma 3. *For a specific chain $\{j_1, \ldots .j_n\}$ run on M_i, total weight assigned to, is at most $4w(j_n)$.*

Proof. Assume $j_k \in \{j_1, \ldots, j_n\}$ and set of jobs $\{j'_1, \ldots, j'_f\}$ are started by OPT on M_i in time interval $[r(j_k), d(j_k))$. It is clear that $r(j_k) + p(j_k) > r(j'_d) + p(j'_d)$ for $1 \leq d \leq f-1$, thus according to the step 3, ALG would certainly execute jobs $\{j'_1, \ldots, j'_{f-1}\}$, which means weight of no ones would be assigned to the chain. Instead, they would certainly be assigned to themselves. As regards j'_f, in case of not running by ALG, inequation $w(j'_f) < [(\sum_{d=1}^{k} w(j_d)) - w(H_k)] \leq 2w(j_k)$ will be held, in which $w(j'_f)$ will be assigned to the chain. Further more, consider j' is run by OPT and not by ALG during executing of heap $\{j_l, \ldots, j_h\}$. According to the lemma 2, j' is only job with this circumstance and also it is clear that $w(j') < 2w(j_h)$. Hence the total weight assigned to the chain is at most $2[(\sum_{k=1}^{n} w(j_k)) - w(H_n)] \leq 4w(j_n)$. □

For a specific chain $\{j_1, \ldots, j_n\}$ that may contains some heaps, we define a reduced-chain $\{j'_1, \ldots, j'_{n'}\}$ which contains only jobs of the chain that are not within a heap or are one of the m last job of a heap, so that any heap in a reduced-chain has at most m jobs (m refers to the number of machines). Likewise, we define pseudo-chain $\{j''_1, \ldots, j''_n\}$ which contains all jobs of the reduced-chain as well as some pseudo jobs so that any job in the pseudo-chain is within a m-heap (a heap containing exactly m jobs).

There are two steps for constructing pseudo-chain related to a chain. Firstly, only keep any job of the chain that is contained in the related reduced-chain, and then drop the rest job of the chain. Secondly, generate some pseudo jobs for the chain and adjust their size and release time in a way that any remaining job of the chain would be included in a m-heap.

In the second step pseudo jobs should be manually adjust such that they always take part as former jobs of a m-heap. For example, consider j that is not in a heap. It is required to generate $m-1$ pseudo jobs in a way that they plus j form a m-heap in which

j is the last job of the heap. In the other case, consider a heap in a chain that contains less than m jobs. In this case, some pseudo jobs would be generated to extend the heap as a m-heap and also they would be adjusted so that pseudo jobs become former and lowest weighted jobs of the m-heap.

Observation 4. *Consider a chain $\{j_1, \ldots, j_n\}$ with its' related reduced-chain $\{j'_1, \ldots, j'_{n'}\}$ and pseudo-chain $\{j''_1, \ldots, j''_n\}$. It is clear that the following equations will be held.*

$$\sum_{i=1}^{n'} w(j'_i) \leq \sum_{d=1}^{n''} w(j''_d) \quad , \quad j_n = j'_{n'} = j''_{n''}$$

Lemma 4. *For a pseudo-chain $\{j''_1, \ldots, j''_{n''}\}$ we have $\sum_{d=1}^{n''} w(j''_d) < (2m)w(j''_{n''})$.*

Proof. We know pseudo-chain is a sequence of heaps $\{h_1, \ldots, h_z\}$ and for $1 \leq i \leq z$, h_i contains m jobs. $h_i(k)$ Denotes k^{th} job of h_i (e.g., $h_z(m) = j''_{n''}$).
By the step 2 in ALG it is clear that

$$w(j''_{n''}) = w(h_z(m)) > \sum_{i=1}^{z-1} w(h_i(m))$$

Note: $w(j''_{n''}) = w(j_n)$ and $\sum_{i=1}^{z-1} w(h_i(m)) = (\sum_{d=1}^{n-1} w(j_d)) - w(H_{n-1}) < w(j_n)$
We know that $w(h_i(k+1)) > w(h_i(k))$. Therefore:

$$\begin{aligned} & w(j''_{n''}) > \sum_{i=1}^{z-1} w(h_i(1)) \\ + \quad & w(j''_{n''}) > \sum_{i=1}^{z-1} w(h_i(2)) \\ & \vdots \\ + \quad & w(j''_{n''}) > \sum_{i=1}^{z-1} w(h_i(m)) \end{aligned}$$

$$(m)w(j''_{n''}) > \sum_{k=1}^{m} \sum_{i=1}^{z-1} w(h_i(k)) = \sum_{d=1}^{n''-m} w(j''_d)$$

We also know $w(j''_{n''})$ is greater than weight of any job in pseudo-chain, so we have

$\sum_{d=n''-m+1}^{n''} w(j''_d) \leq (m)w(j''_{n''})$ hence:

$$\sum_{d=1}^{n''-m} w(j''_d) + \sum_{d=n''-m+1}^{n''} w(j''_d) = \sum_{d=1}^{n''} w(j''_d) < (2m)w(j''_{n''})$$

□

Lemma 5. *For a chain $\{j_1 \ldots, j_n\}$ running on one of the m machines, if both ALG and OPT schedule j_k, then it will be contained within the related reduced-chain (for $1 \leq k \leq n$).*

Proof. Assume j_k is not within the related reduced-chain. In this case, there would be a heap with more than m jobs, which j_k is one of the former jobs of the heap (it is released sooner than m last job of the heap). Now we show that scheduling j_k by OPT is in direct contradiction to optimal solution. Since OPT is able to execute at most m jobs of a heap (because of overlapping of jobs and similarity of machines), if OPT runs j_k, definitely there will be a job j' within the last m jobs of the heap which will be omitted by OPT. In addition, according to properties of heaps, we have $w(j') > w(j_k)$ and $p(j') < p(j_k)$ and $r(j') \geq r(j_k)$ and $r(j') + p(j') \leq r(j_k) + p(j_k)$. Therefore, we can introduce better solution, which schedules same jobs as OPT except j_k that is replaced by j'. □

Theorem 2. *For $m > 1$, the competitive ratio of the ALG is at most $2m + 4$ for D-benevolent instances.*

Proof. Since the *ALG* only completes the last job of the each chain and weight of all jobs in OPT are assigned either to the chains or jobs of the chains, it is sufficient to compare total weight assigned to a specific chain $\{j_1, \ldots, j_n\}$ and jobs of it, with weight of last job of the chain $w(j_n)$ (the only job in the chain that *ALG* completes it). By lemma 3, the total weight assigned to the chain is at most $4w(j_n)$, also according to the weight assignment rules, the total weight assigned to a specific job j_k is at most $w(j_k)$, in which j_k is within the related reduced-chain $\{j'_1, \ldots, j'_{n'}\}$, by lemma 5. Now consider that sequence $\{j''_1, \ldots, j''_{n''}\}$ is the related pseudo-chain of the chain. In this case, total weight assigned to jobs of the chain is at most $\sum_{d=1}^{n'} w(j'_d) \leq \sum_{d}^{n''} w(j''_d) < (2m)w(j''_{n''}) = (2m)w(j_n)$.
Hence:

$$(2m)w(j_n) + 4w(j_n) = (2m + 4)w(j_n)$$

□

4 Acknowledgment

This work was supported by Tehran University. Our special thanks go to the University of Tehran, College of Engineering and Department of Engineering Science for providing

all the necessary facilities available to us for successfully conducting this research.

References

[1] Awerbuch B, Bartal Y, Fiat A, Rosén A (1994) Competitive non-preemptive call control. In: SODA, vol 94, pp 312–320

[2] Canetti R, Irani S (1998) Bounding the power of preemption in randomized scheduling. SIAM Journal on Computing 27(4):993–1015

[3] Carlisle MC, Lloyd EL (1991) On the k-coloring of intervals. In: Advances in Computing and InformationICCI'91, Springer, pp 90–101

[4] Epstein L, Jez L, Sgall J, van Stee R (2012) Online scheduling of jobs with fixed start times on related machines. In: Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, Springer, pp 134–145

[5] Faigle U, Nawijn WM (1995) Note on scheduling intervals on-line. Discrete Applied Mathematics 58(1):13–17

[6] Fung SP, Poon CK, Zheng F (2014) Improved randomized online scheduling of intervals and jobs. Theory of Computing Systems 55(1):202–228

[7] Woeginger GJ (1994) On-line scheduling of jobs with fixed start and end times. Theoretical Computer Science 130(1):5–16